IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Serial No. 09/884,009
Filing Date ..... June 18, 2001
Inventorship ..... Hsieh
Appellant..... Microsoft Corporation
Group Art Unit 2194
Examiner ..... Wu, Quing Yuan
Attorney's Docket No. MS1-749US
Title: Systems and Methods for Managing a Run Queue

APPEAL BRIEF RESPONSIVE TO THE NOVEMBER 16, 2005 FINAL OFFICE ACTION

To: Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-14503

From: Brian G. Hart (Tel. 509-324-9256; Fax 509.323-8979)
Lee & Hayes, PLLC
421 W. Riverside Avenue, Suite 500
Spokane, WA 99201

Introduction

Pursuant to 37 C.F.R. § 41.37, Appellant hereby submits an appeal brief within two months of the requisite time from the date of filing the Notice of Appeal, which was filed on February 22, 2006. Appellant appeals to the Board of Patent Appeals and Interferences seeking review of rejections to subject matter in the above captioned patent application. These rejections result from a Final Office Action dated November 16, 2005 (hereinafter referred to as the "Final Action").

REAL PARTY IN INTEREST

The real party in interest is Microsoft Corporation, the assignee of all right, title and interest in and to the claimed subject matter.

RELATED APPEALS AND INTERFERENCES

Appellant is not aware of any other appeals, interferences, or judicial proceedings that will directly affect, be directly affected by, or otherwise have a bearing on the Board's decision to this pending appeal.

STATUS OF CLAIMS

Claims 1-24 were originally pending. Claims 1-6, 8-11, 13-21, and 23 were amended. Claims 7, 12, 22, and 24 were canceled without prejudice. No claims were added. Accordingly, claims 1-6, 8-11, 13-21, and 23 are pending. The pending claims are presented in the Appendix of Appealed Claims on page 18.

Claims 1-6, 8-11, 13-21, and 23 stand rejected under 35 USC §103(a). Appellant appeals the rejections of these claims.

STATUS OF AMENDMENTS

Appellant has not amended the claims subsequent to the date of the Final Action.

SUMMARY OF CLAIMED SUBJECT MATTER

This summary section provides a concise explanation of each of the independent claims, including specific reference characters and reference to the specification. These specific reference characters are examples of particular elements of the drawings for certain claimed embodiments. It is understood that the claims are not to be limited to solely the elements corresponding to these reference characters and that this section is provided to comply with the requirement of 37 CFR § 41.37(c)(1)(v).

Independent **claim 1** recites in part:

- "[a] method to be implemented in a computer system comprising a processor and a memory"
- "managing a run queue comprising a first plurality of threads sorted with respect to one another based on thread priority", and
- "in a deterministic amount of time equivalent to an amount of time to insert a single thread into the run queue, associating a second plurality of threads that is priority sorted with the run queue in a manner that maintains a priority based scheduling semantic of the run queue."

An exemplary computing device 1410 for implementing the "method to be implemented in a computer system comprising a processor and a memory" is described on pages 25 through 30 of the specification with respect to Fig. 14. An exemplary "run queue comprising a first plurality of threads sorted with respect to one another based on thread priority" is described at page 21, line 6, through page 24, line 5, with respect to the "run queue" 1100 of Fig. 11. An exemplary

procedure to "in a deterministic amount of time equivalent to an amount of time to insert a single thread into the run queue, associating a second plurality of threads that is priority sorted with the run queue in a manner that maintains a priority based scheduling semantic of the run queue", as claim 1 recites, is described at pages 24-24 of the specification with respect to Figs. 12 and 13.

Independent **claim 8** recites in part:

- "[a] system",
- "a run queue [...] comprising a first plurality of threads, each thread in the first plurality of threads having a respective priority, the first plurality of threads being sorted such that a thread having a high priority is removed from the run queue before a thread having a lower priority",
- "in an amount of time to insert a single thread into the run queue, associating the second plurality of threads that is priority sorted with the run queue, the associating maintaining a priority based scheduling semantic of the run queue."

Independent **claim 16** recites in part:

- [a] computer-readable storage medium comprising computer-program instructions to manage a run queue of executable threads sorted with respect to one another based on thread priority",
- "in a deterministic amount of time that is independent of the number of threads in a second plurality of threads that is priority sorted, the deterministic amount of time being a time to insert a single thread into the run queue, associating the second plurality of threads with a first plurality of threads in the run queue in a manner that maintains a priority based scheduling semantic of the run queue."

An exemplary "computer-readable storage medium comprising computer-program instructions to manage a run queue of executable threads sorted with respect to one another based on thread priority" is described on page 27, line 8, through page 28, line 16, with respect to component 1414 of Fig. 14. An exemplary "run queue of executable threads sorted with respect to one another based on thread priority" is described at page 21, line 6, through page 24, line 5, with respect to the "run queue" 1100 of Fig. 11. Exemplary operations for "in a deterministic amount of time that is independent of the number of threads in a second plurality of threads that is priority sorted, the deterministic amount of time being a time to insert a single thread into the run queue, associating the second plurality of threads with a first plurality of threads in the run queue in a manner that maintains a priority based scheduling semantic of the run queue", as claim 16 recites, is described at pages 24-24 of the specification with respect to Figs. 12 and 13.

Independent **claim 23** recites in part:

- "[a] computer-readable medium comprising computer-program instructions executable by a processor",
- "managing a run queue with a run queue data structure, the run queue data structure comprising:
    - a first dimension data field comprising a first plurality of threads sorted with respect to thread priority; and
    - a second dimension data field comprising a second plurality of threads sorted based on thread priority, the second plurality of threads comprising a root thread and one or more other threads."

An exemplary "computer-readable medium comprising computer-program instructions executable by a processor" is described on page 27, line 8, through page 28, line 16, with respect to component 1414 of Fig. 14. An exemplary "a run queue [...] datat structure comprising: a first dimension data field [and] a second dimension data field" is described at page 21, line 6, through page 24, line 5, with respect to the "run queue" 1100 of Fig. 11.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The following grounds of rejection are for review on appeal. Claims 1-6, 8-11, 13-21, and 23 stand rejected in the Final Action under 35 USC §103(a) as being unpatentable over US patent number 6,609,161 to Young.

ARGUMENT

35 USC §103(a) Rejections

Claims 1-6, 8-11, 13-21, and 23 stand rejected under 35 USC §103(a) as being unpatentable over US patent number 6,609,161 to Young. Appellant respectfully traverses these rejections.

Claim 1 recites

- "the method for managing a run queue comprising a first plurality of threads sorted with respect to one another based on thread priority", and
- "in a deterministic amount of time equivalent to an amount of time to insert a single thread into the run queue, associating a second plurality of threads that is priority sorted with the run queue in a manner that maintains a priority based scheduling semantic of the run queue."

In addressing these features of claim 1 and Appellant's previous discussed reasons why Young did not teach or suggest these claimed features (presented in the response dated August 24, 2005), the Final Action at page 6 in combination with the Advisory Action argues: (1) that **the broadest reasonable interpretation of "run queue" is where** "programs/processes/threads/**commands are taken from the head of the queue to be executed or process[ed]"; and (2) "and 'a command' as defined by The Authoritative Dictionary of IEEE Standard Terms Seventh Edition [is] 'an instruction in machine language'**". In view of these assertions, the Action asserts that Young meets the limitation of "run queue" and claim 1 is obvious over Young. Appellant respectfully disagrees. Young does not teach or suggest programs/processes in a run queue. And, with respect to

commands, Young teaches only **rewind, write, compare, verify, or other SCSI commands. A SCSI command is not "an instruction in machine language". And, it is well-known in the art that a "run queue" as claim 1 requires can not be used to store SCSI commands**, as the Action asserts.

During examination plain meaning is given to a claimed term unless the specification provides meaning for the term, whereupon which the specification must be used to identify the meaning ascribed to the term by the inventor. (MPEP §2111.01). **In this case, the Action has not used either the plain meaning of the claimed term "run queue" or the explicit definition provided by Appellant's specification.** Instead the Action changes the plain and well-known meaning typically associated with a "run queue" by substituting its own interpretation that is completely contrary to the meaning that one of ordinary skill in the art at the time of invention would have given the term, and also completely contrary to what is described in the specification for the term. (The explicit teachings of Young to not even use the phrase "run queue" or "run" anything).

It is well-known in the art that a "run queue", as claim 1 requires, is not for storing rewind, write, compare, verify, or other SCSI commands, as the Action asserts. A "run queue" is for storing threads representing respective paths of execution through a computer-program (process) for execution by a computer. Not only was this plain meaning of the use of a "run queue" well-known at the time of invention, but it is also the use of a "run queue" described in Appellants specification. The Background section of Appellant's specification at page 2, lines 1-2 clearly states that threads are stored in a "run queue for subsequent execution." Page 1 of the Background sections clearly states that "[a] thread is

basically a path of execution through a computer program application." It is well-known that a path of execution through a computer program may be represented with machine language commands. It is also well known that the SCSI commands that Young stores in a queue are not machine language commands and SCSI commands do not represent "a path of execution through a computer program application". Young does not even teach or suggest such things.

Young teaches a two-dimensional **SCSI command block (SCB) execution queue to deliver multiple SCSI commands to a target such as a SCSI/peripheral device** (please see the Abstract). Young, at column 2, lines 63-64, explicitly describes that "[e]ach command block includes a command for target device". Young describes at column 1, lines 12- 16, that an example of such a control block is a SCSI command block (SCB) used to transfer information between a software host adapter bus driver and a peripheral device. Examples of SCSI commands include, for example, rewind, write, compare, verify, etc.

In view of these well-known plain meanings of a run queue, the specification supported descriptions of a "run queue", and Young's explicit teachings, the Action's assertion that the claimed "run queue comprising a plurality of threads" is the same as Young's command queue used to store SCSI commands, is not a reasonable assertion. Even Young is aware of this difference, since Young teaches that SCSI commands are stored in a SCSI Command Block Execution Queue – not "a run queue", as claim 1 requires. The explicit teachings of Young to not even use the phrase "run queue" or "run" anything.

Appellant respectfully submits that if Young tried to insert such SCSI rewind, write, compare, verify, or other commands (meant to be parsed by a SCSI

compliant peripheral device) into a "run queue" as claim 1 requires, such insertion would likely harm runtime operation of any system that relied on the run queue to store threads representing a path of execution through a computer-program process. In such an illogical scenario, the system's thread scheduling mechanism would not encounter a thread, but would instead encounter a rewind, write, compare, verify, or other type of SCSI command. Clearly this command does not belong in a "run queue", but instead as Young teaches, this SCSI command is for sending to a target peripheral device for parsing. In view of this, Young does not teach a "run queue comprising a plurality of threads". Rather, Young teaches a SCSI command queue comprising SCSI commands for subsequent communication to a peripheral device.

In view of the above plain and well-known meaning of the purpose of a "run queue", Appellant is not attempting to read limitations of the specification into the claims for purposes of avoiding prior art. Instead, Appellant is merely relying on a fundamental aspect of patent law that dictates that claimed subject matter cannot be examined in a vacuum. As §2111.01 states, a term must be examined in view of plain meaning unless the specification provides meaning for the term, whereupon which the specification must be used to identify the meaning ascribed to the term by the inventor. In this case, **the Action has assigned a meaning to a term that is contrary not only to plain and well-known meaning for the term, but also different than meaning given to the term by the inventor**. In light of this, the Action is seemingly relying on personal knowledge to support this otherwise unsupported meaning ascribed to a claim feature.

According to 37 CFR §1.104(d)(2), "[w]hen a rejection in an application is based on facts within the personal knowledge of an employee of the office, the data shall be as specific as possible, and the reference must be supported, when called for by the Appellant, by the affidavit of such employee, and such affidavit shall be subject to contradiction or explanation by the affidavits of the Appellant and other persons." In view of this, and regardless of whether the form of the Actions' rejection of claim 1 is proper under MPEP §706.02(j), if this rejection is maintained on a similar basis in a subsequent action, the Examiner is again requested to supply such an affidavit to support this otherwise unsupported modification to the SCSI command queue of Young. Otherwise, and without additional support, it is respectfully submitted the Final Action's conclusion does not represent the conclusion of a person of ordinary skill at the time of invention, and thereby, cannot represent a "broadest reasonable interpretation" of a "run queue", as the Final Action asserts.

For these reasons alone, Young does not teach or suggest each and every element of claim 1.

Withdrawal of the 35 USC §103(a) rejection of claim 1 is requested.

Additionally, claim 1 includes further features that are not taught or suggested by Young. For example, claim 1 also requires "threads". When addressing this feature, the Final Action at page 7 points out that the term "thread", as defined by Microsoft Computer Dictionary Fifth Addition, is "a process that is part of a larger process or program". Given this definition, the Action asserts that the "broadest reasonable interpretation" of a "thread" is met by "a SCSI command block". Appellant respectfully submits that this broad

interpretation is clearly not reasonable, especially in light of the explicit definition provided by the referenced dictionary, which defines a thread as "a process [...]". Although a command of machine language can be part of a computer-executable "process", it is well-known that a SCSI command is not a machine language command, and it is well-known that a SCSI command is not a thread of execution through a "process".

Young, at column 2, lines 63-64, teaches that "[e]ach command block includes a command for target device". Young describes at column 1, lines 12-16, that it example of such a control block is a SCSI command block (SCB) used to transfer information between a software host adapter bus driver and a peripheral device. **Examples of SCSI commands include, for example, rewind, write, compare, verify, etc**. In contrast to SCSI commands, it is well-known in the art of computer-science and computer programming that a thread within the context within which it is used (i.e., "part of a larger process or program") is a part of a computer-program application that can execute independently of other parts of the computer-program application. In contrast to "threads", SCSI commands cannot execute independently of other parts of a computer-program application **because SCSI commands are not representative of computer program language paths of execution**. Instead, SCSI commands are blocks of information that upon being parsed, direct peripheral devices to perform some action such as rewind, verify, compare, etc. In view of these express teachings of Young, a "SCSI command block" is clearly not "**a process**", as required by the definition of a thread provided by the Final Action. Since a SCSI command is not a process, a SCSI command cannot be "a process that is part of a larger process or program".

In view of the above, Young's express disclosure of a SCSI command does not meet the definition of a thread that was provided by the Final Action. Appellant respectfully submits that if Young tried to insert a SCSI command into a system that typically stores threads into a "run queue", the inserted SCSI command block would at least temporarily incapacitate the runtime processing operations of the system. This is because SCSI commands such as rewind, write, verify, compare and other SCSI commands do not represent a path of execution in a computer program (i.e., a thread). In such an unlikely scenario, a thread scheduling mechanism would not encounter a thread as typically expected, but would instead find a SCSI command block of information for a target peripheral device.

This is another example of where the Office is seemingly using personal knowledge to modify the well-known and plain meaning of "threads" that are stored in a "run queue" by substituting SCSI commands for the threads. Not only are these modifications unsupported and contrary to the well-known and plain meaning of the claim term within its presented context, but also contrary to the clear descriptions of the term in Appellant's specification.

Again, Appellant is not attempting to read limitations of the specification into the claims for purpose of avoiding prior art. Instead the Appellant is merely relying on the Office to follow the directions of MPEP §2111.01 and not examine the claims in a vacuum. Here, the Final Action has assigned meaning to "threads" that is not only contrary to plain meaning and what would have normally been ascribed to the term by a person of ordinary skill in the art at the time of invention, but also contrary to the written description of "threads" in Appellant specification.

Thus, the Actions broad interpretation of the claimed term "threads" as being the same as SCSI commands is not reasonable and not representative of the meaning of the claimed feature of "threads", as claim 1 requires.

For these additional reasons, Young does not teach or suggest each and every element of claim 1. Withdrawal of the 35 USC §103(a) rejection of claim 1 is requested.

Claims 2-6 depend from claim 1 and are allowable over Young solely by virtue of this dependency. Accordingly, withdrawal of the 35 USC §103(a) rejection of claims 2-6 is requested.

Claim 8 recites

- "[a] system for managing a run queue, the run queue comprising a first plurality of threads, each thread in the first plurality of threads having a respective priority, the first plurality of threads being sorted such that a thread having a high priority is removed from the run queue before a thread having a lower priority", and
- "in an amount of time to insert a single thread into the run queue, associating the second plurality of threads that is priority sorted with the run queue, the associating maintaining a priority based scheduling semantic of the run queue."

For the reasons already discussed, Young does not teach or suggest these claimed features.

Withdrawal of the 35 USC §103(a) rejection of claim 8 is requested.

Claims 9-11 and 13-15 depend from claim 8 and are allowable over Young solely by virtue of this dependency. Accordingly, withdrawal of the 35 USC §103(a) rejection of claims 9-11 and 13-15 is requested.

Claim 16 recites

- "computer-program instructions to manage a run queue of executable threads sorted with respect to one another based on thread priority", and
- "in a deterministic amount of time that is independent of the number of threads in a second plurality of threads that is priority sorted, the deterministic amount of time being a time to insert a single thread into the run queue, associating the second plurality of threads with a first plurality of threads in the run queue in a manner that maintains a priority based scheduling semantic of the run queue."

For the reasons already discussed, Young does not teach or suggest these claimed features.

Withdrawal of the 35 USC §103(a) rejection of claim 16 is requested.

Claims 17-21 depend from claim 16 and are allowable over Young solely by virtue of this dependency. Accordingly, withdrawal of the 35 USC §103(a) rejection of claims 17-21 is requested.

Claim 23 recites

- "managing a run queue with a run queue data structure, the run queue data structure comprising: a first dimension data field comprising a first plurality of threads sorted with respect to thread priority", and
- "a second dimension data field comprising a second plurality of threads sorted based on thread priority, the second plurality of threads comprising a root thread and one or more other threads."

For the reasons already discussed, Young does not teach or suggest these claimed features.

Withdrawal of the 35 USC §103(a) rejection of claim 23 is requested.

Conclusion

Appellant respectfully submits that the rejections to the pending claims have been traversed. The pending claims are in condition for allowance and action to that end is urgently requested.

Respectfully Submitted,

Dated: 9/24/2006 By: 

Brian G. Hart
Lee & Hayes, PLLC
Reg. No. 44,421
(509) 324-9256

APPENDIX OF APPEALED CLAIMS

1. (Previously presented) A method to be implemented in a computer system comprising a processor and a memory, the method for managing a run queue comprising a first plurality of threads sorted with respect to one another based on thread priority, the method comprising:

in a deterministic amount of time equivalent to an amount of time to insert a single thread into the run queue, associating a second plurality of threads that is priority sorted with the run queue in a manner that maintains a priority based scheduling semantic of the run queue.

2. (Previously presented) A method as recited in claim 1, wherein the second plurality of threads comprises a root thread, and wherein associating the second plurality of threads with the run queue further comprises inserting only the root thread into the run queue.

3. (Previously presented) A method as recited in claim 1, wherein the associating the second plurality of threads with the run queue further comprises inserting each thread in the second plurality of threads into the run queue independent of any additional other queue access.

4. (Previously presented) A method as recited in claim 1, wherein associating the second plurality of threads with the run queue further comprises inserting only a root thread of the second plurality of threads into the run queue.

5. (Previously presented) A method as recited in claim 1, wherein associating the second plurality of threads with the run queue further comprises:

inserting only a root thread of the second plurality of threads into the run queue; and

wherein the method further comprises:

removing the root thread from the run queue; and

responsive to removing the root thread, inserting a next thread of the second plurality of threads into the run queue such that the priority based scheduling semantic of the run queue is preserved.

6. (Previously presented) A method as recited in claim 1, wherein the method further comprises:

inserting a root thread of the second plurality of threads into the run queue;

removing the root thread from the run queue for execution; and

responsive to removing the root thread and independent of any additional other queue access, inserting a next thread of the second plurality of threads into the run queue.

7. (Canceled).

8. (Previously presented) A system for managing a run queue, the run queue comprising a first plurality of threads, each thread in the first plurality of

threads having a respective priority, the first plurality of threads being sorted such that a thread having a high priority is removed from the run queue before a thread having a lower priority, the system comprising:

a memory for storing the run queue and computer-executable instructions;

a processor operatively coupled to the memory, the processor being configured to execute the computer–executable instructions for:

in an amount of time to insert a single thread into the run queue, associating the second plurality of threads that is priority sorted with the run queue, the associating maintaining a priority based scheduling semantic of the run queue.

9. (Previously presented) A system as recited in claim 8, wherein associating the second plurality of threads with the run queue is performed independent of more than a single other queue access.

10. (Previously presented) A system as recited in claim 8, wherein the second plurality of threads comprises a root thread operatively coupled to one or more other threads of the second plurality of threads, each of the one or more other threads having a respective priority that is a lower priority or an equal priority as compared to a priority of the root thread.

11. (Previously presented) A system as recited in claim 8, wherein associating the second plurality of threads with the run queue further comprises inserting only a root thread of the second plurality of threads into the run queue.

12. (Canceled).

13. (Previously presented) A system as recited in claim 8:

wherein the first plurality of threads is a first linked list data structure;

wherein the second plurality of threads is a second linked list data structure comprising a root node that is operatively coupled to one or more other threads in the second plurality of threads; and

wherein the single insert operation is an operation comprising inserting the root node into a position in the first linked list data structure.

14. (Previously presented) A system as recited in claim 8, wherein associating the second plurality of threads with the run queue further comprises:

inserting only a root thread of the second plurality of threads into the run queue; and

wherein the method further comprises:

removing the root thread from the run queue; and

responsive to removing the root thread, inserting a next thread of the second plurality of threads into the run queue such that a priority based scheduling semantic of the run queue is preserved.

15. (Previously presented) A system as recited in claim 8, wherein the processor is further configured to execute computer program instructions for:

inserting a root thread of the second plurality of threads into the run queue;

removing the root thread from the run queue for execution; and

responsive to removing the root thread and independent of any additional other queue access, inserting a next thread of the second plurality of threads into the run queue.

16. (Previously presented) A computer-readable storage medium comprising computer-program instructions to manage a run queue of executable threads sorted with respect to one another based on thread priority, the computer-program instructions being executable by a processor for:

in a deterministic amount of time that is independent of the number of threads in a second plurality of threads that is priority sorted, the deterministic amount of time being a time to insert a single thread into the run queue, associating the second plurality of threads with a first plurality of threads in the run queue in a manner that maintains a priority based scheduling semantic of the run queue.

17. (Previously presented) A computer-readable storage medium as recited in claim 16, wherein the second plurality of threads comprises a root thread that is operatively coupled to one or more other threads of the second plurality of threads, and wherein the computer-program instructions for associating further comprise instructions for inserting only the root thread into the first plurality of threads.

18. (Previously presented) A computer-readable storage medium as recited in claim 16, wherein the first plurality of threads is a first linked list data structure, the second plurality of threads is a second linked list data structure comprising a root node that is operatively coupled to one or more other threads in the second plurality of threads, and the deterministic amount of time is a result of a single insert operation to insert the root node into the first linked list data structure.

19. (Previously presented) A computer-readable storage medium as recited in claim 16, wherein the computer-program instructions for associating further comprise instructions for:

inserting only a root thread of the second plurality of threads into the first plurality of threads;

and wherein the computer-program instructions further comprise instructions for:

removing the root thread from the run queue; and

responsive to removing the root thread, inserting a next thread of the second plurality of threads into the first plurality of threads in a manner that maintains a priority based scheduling semantic of the run queue.

20. (Previously presented) A computer-readable storage medium as recited in claim 19, wherein the computer-program instructions for inserting the next thread are performed independent of an other queue.

21. (Previously presented) A computer-readable storage medium as recited in claim 16, wherein the computer-program instructions for associating further comprise instructions for:

inserting a root thread of the second plurality of threads into the first plurality;

removing the root thread from the first plurality of threads for execution; and

responsive to removing the root thread, inserting a next thread of the second plurality of threads into the first plurality of threads independent of any additional access to another different queue.

22. (Canceled).

23. (Previously presented) A computer-readable medium comprising computer-program instructions executable by a processor for:

managing a run queue with a run queue data structure, the run queue data structure comprising:

a first dimension data field comprising a first plurality of threads sorted with respect to thread priority; and

a second dimension data field comprising a second plurality of threads sorted based on thread priority, the second plurality of threads comprising a root thread and one or more other threads.

24. (Canceled).